Sandia National Laboratories

Exceptional service in the national interest

ME469: Common Discretization Approaches: Edge-based,Vertex-Centered (EBVC), and Cell-Centered (CC)

Stefan P. Domino[1,2]

[1] Computational Thermal and Fluid Mechanics, Sandia National Laboratories

[2] Institute for Computational and Mathematical Engineering, Stanford

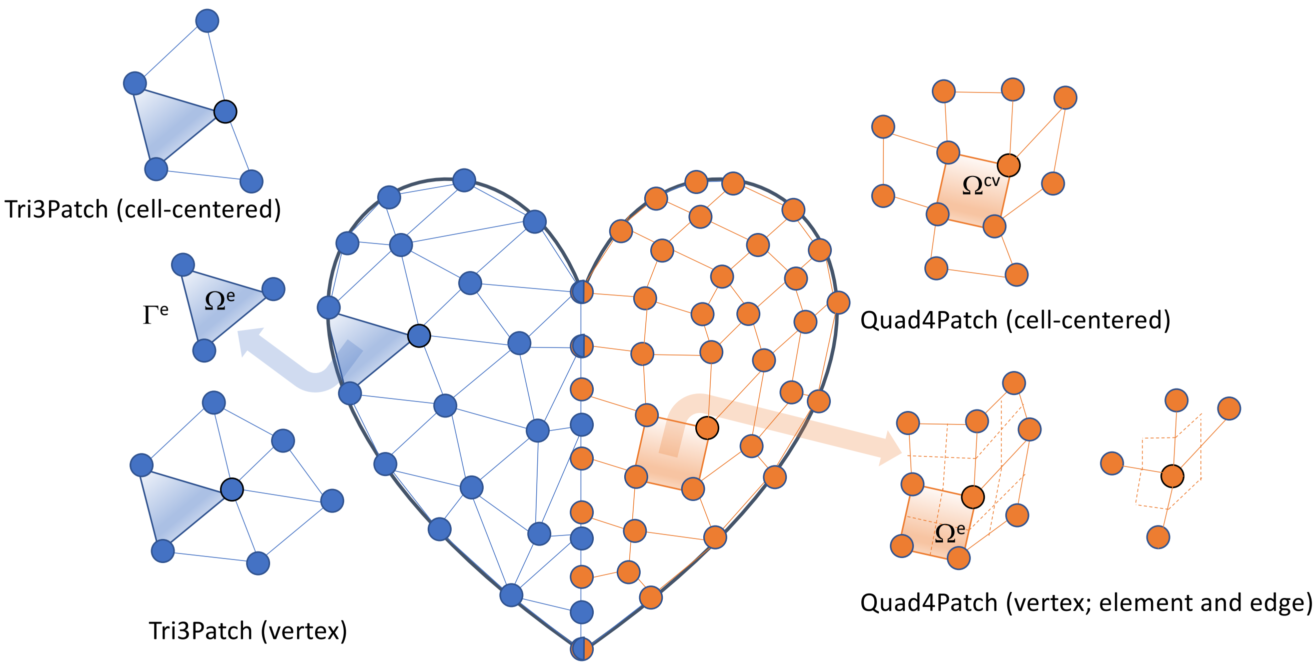
SAND2018-4536 PE

U.S. DEPARTMENT OF ENERGY

NNSA

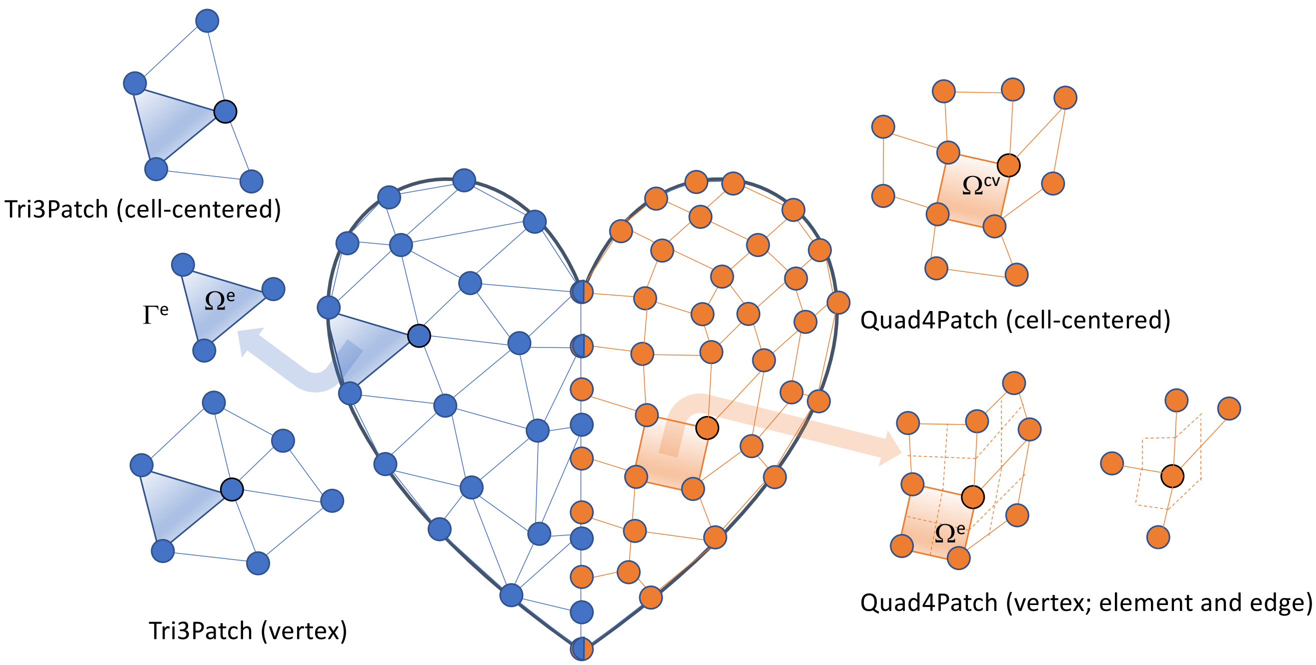
Review of Discretization Options

- Degree-of-freedom (DOF) for:
  - Cell-centered: Stencil is based on a element:face:element
  - DOFs at vertices of elements, or “nodes”, element:node (CVFEM, FEM), edge:node (EBVC)



Review of Discretization Options: New, a nodal-basis...

- Degree-of-freedom (DOF) for:
  - Cell-centered: Stencil is based on a element:face:element
  - DOFs at vertices of elements, or “nodes”, element:node (CVFEM, FEM), edge:node (EBVC)



- Definition of an interpolation function:

$$\phi_{ip} = \sum_n N_n^{ip} \phi_n$$

- N_n^{ip} is the Lagrange function associated with node n
- ϕ_n is the value of the DOF at node n
- The nodal basis functions obey equipartition of unity and satisfy, $N_n^{x_j} = \delta_{nj}$

Fundamentals of Discretization: Surface vs Volume Integrations

- Given a partial differential equation (PDE) and associated volumetric form:

$$\int \frac{\partial F_j}{\partial x_j} dV = \int S dV$$

- Applying Gauss Divergence provides the standard finite volume form for fluxes in surface integral form:

$$\int \frac{\partial F_j}{\partial x_j} dV = \int F_j n_j dS \longrightarrow \int F_j n_j dS = \int S dV$$

- We can also multiple PDE by an arbitrary test function, w, and integrate over a volume,

$$\int w \frac{\partial F_j}{\partial x_j} dV = \int w S dV$$

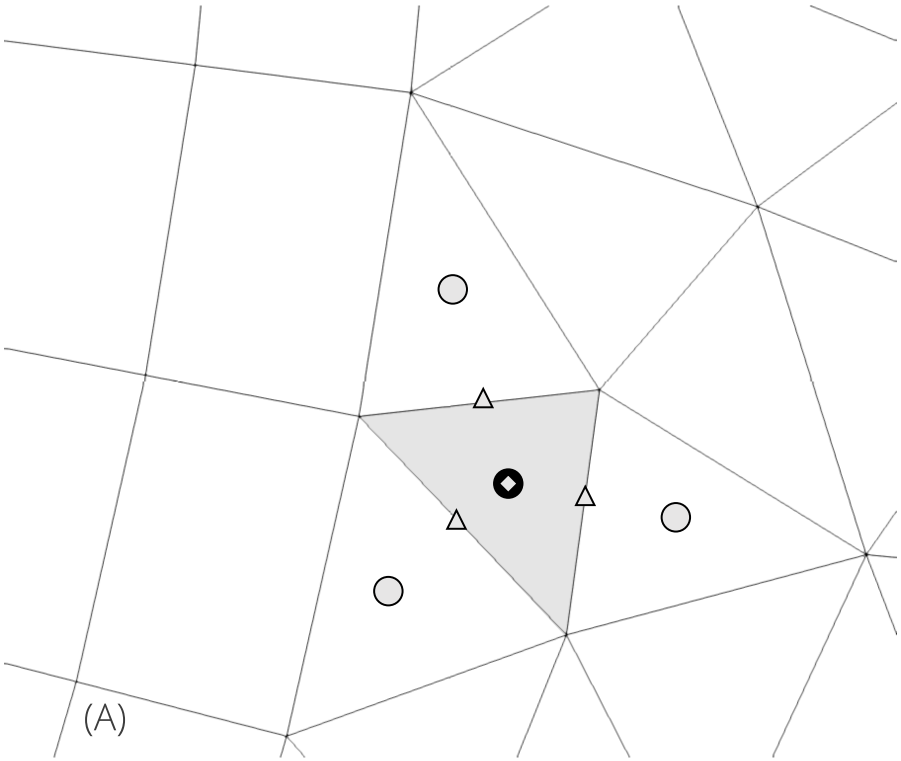
$$\int \frac{\partial w}{\partial x_j} F_j dV - \int w F_j dS = \int w S dV$$

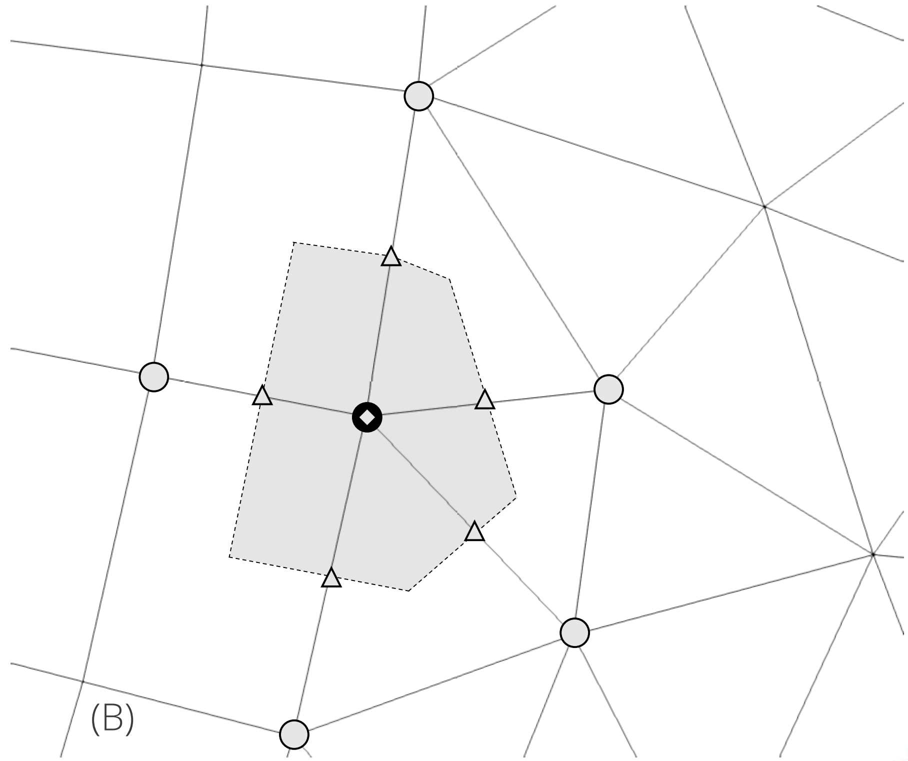
Next, integrate by parts and apply Gauss-Divergence. Note, that test function must be differentiable – shown here, at least once..

$$\frac{\partial w F_j}{\partial x_j} = w \frac{\partial F_j}{\partial x_j} + \frac{\partial w}{\partial x_j} F_j$$

Cell-Centered (CC) and Edge-Based Vertex-Centered (EBVC): Each are generally a two-state scheme, Left (L) and Right(R)

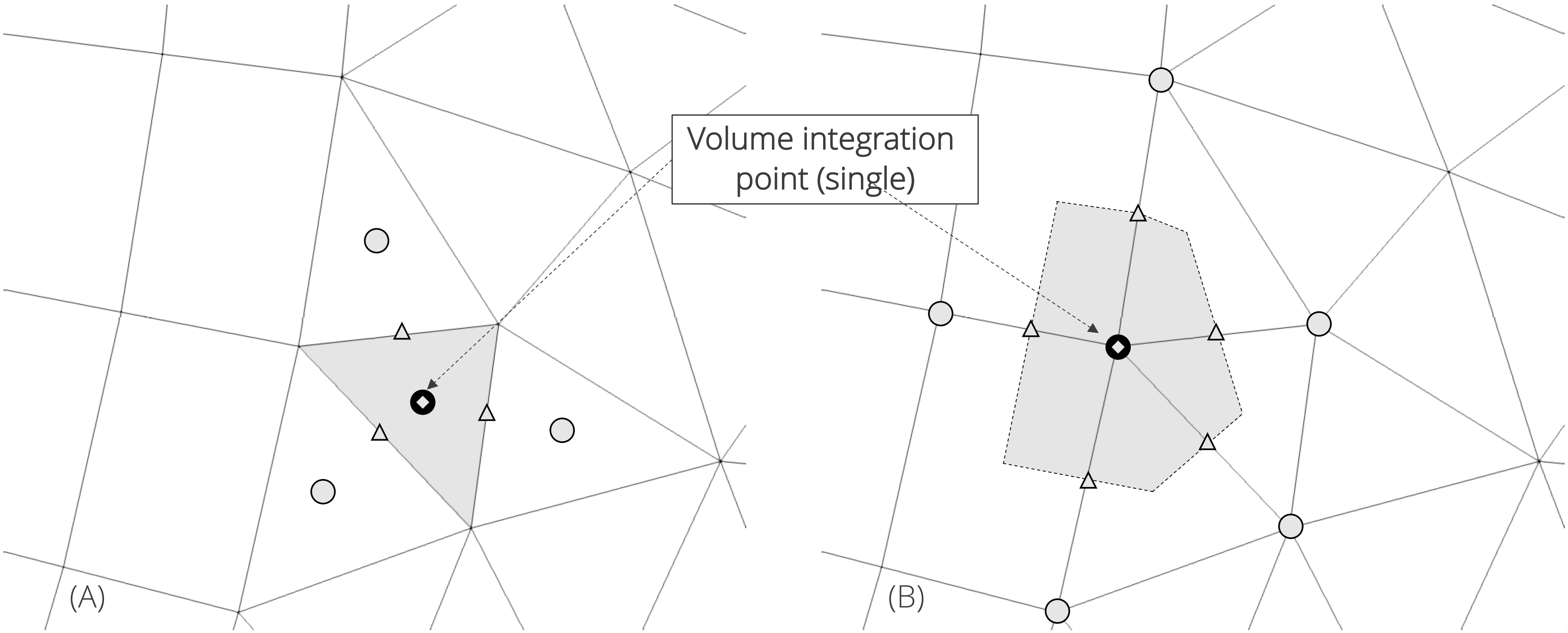
- CC (A) and EBVC (B) – As shown below, each have a set of common features
  - Class Discussion: What are the common features?





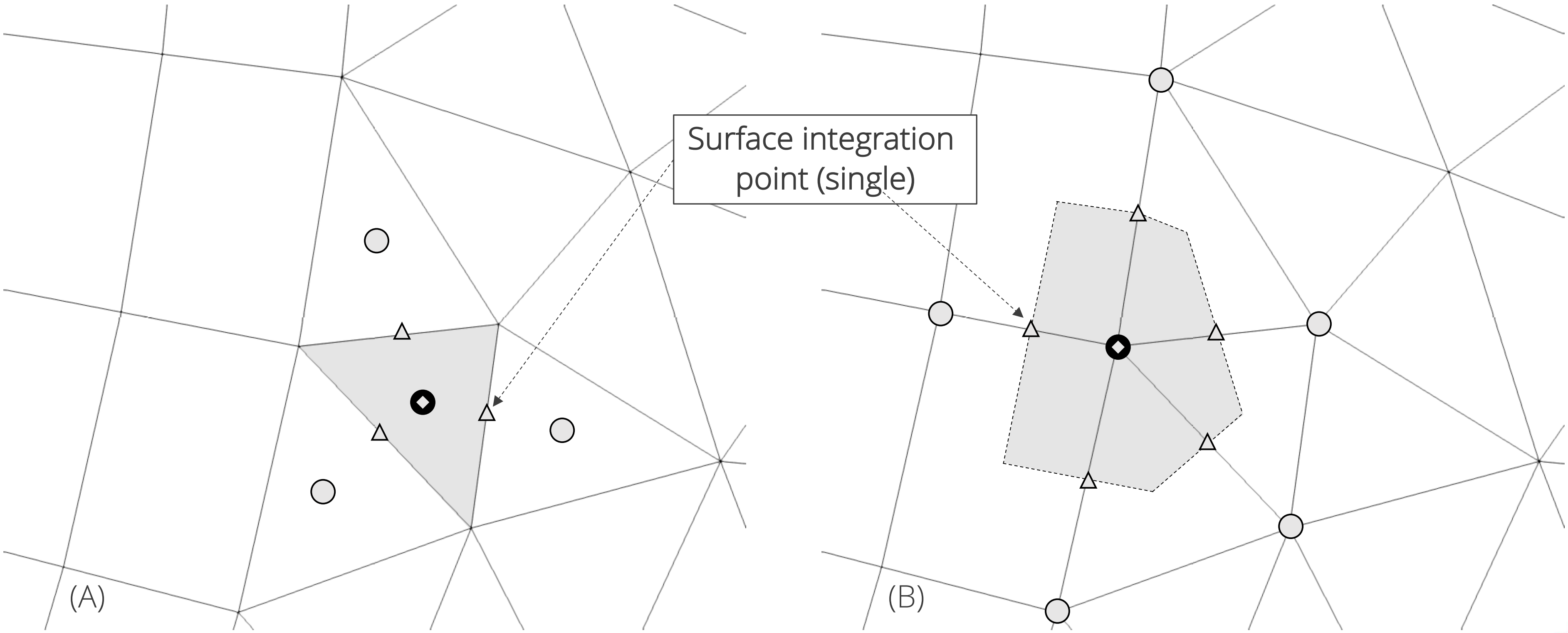
Cell-Centered (CC) and Edge-Based Vertex-Centered (EBVC): Each are generally a two-state scheme, Left (L) and Right(R)

- CC (A) and EBVC (B) – As shown below, each have a set of common features
  - Class Discussion: What are the common features?



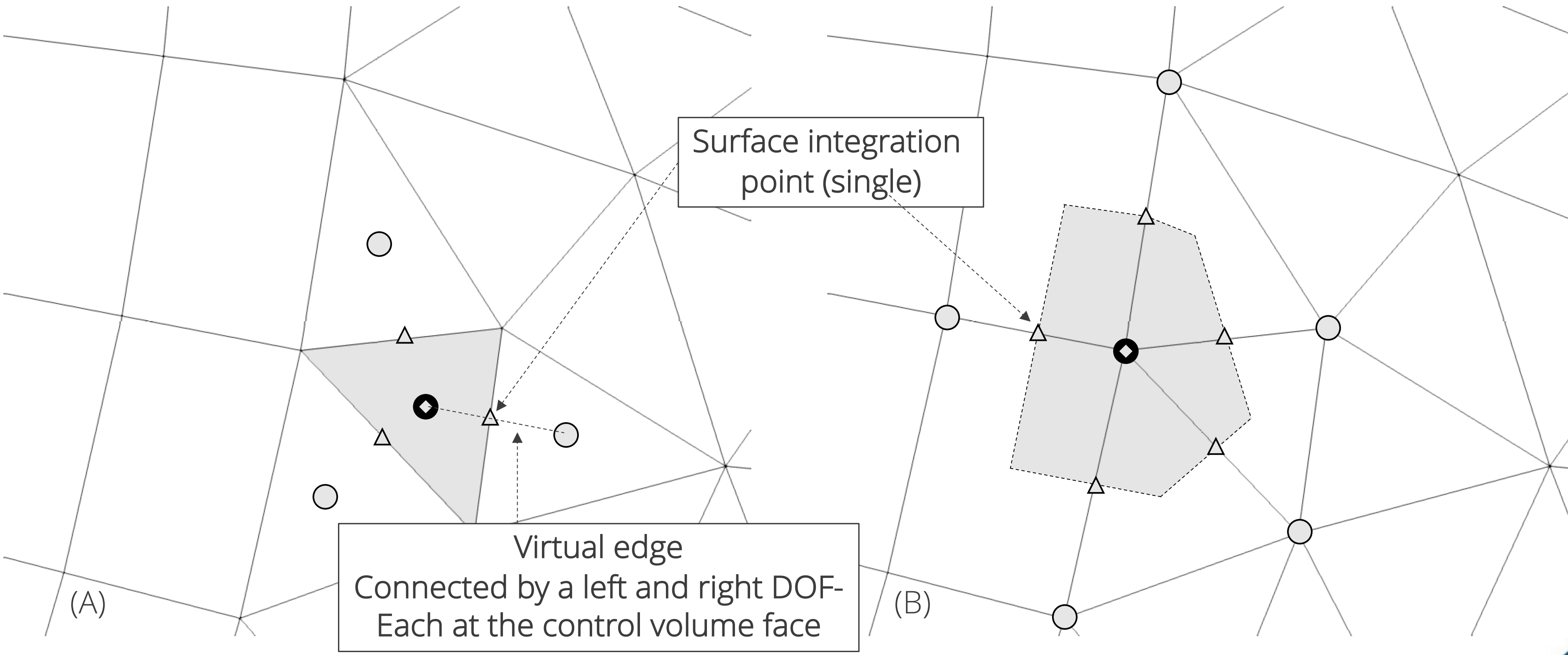
Cell-Centered (CC) and Edge-Based Vertex-Centered (EBVC): Each are generally a two-state scheme, Left (L) and Right(R)

- CC (A) and EBVC (B) – As shown below, each have a set of common features
  - Class Discussion: What are the common features?



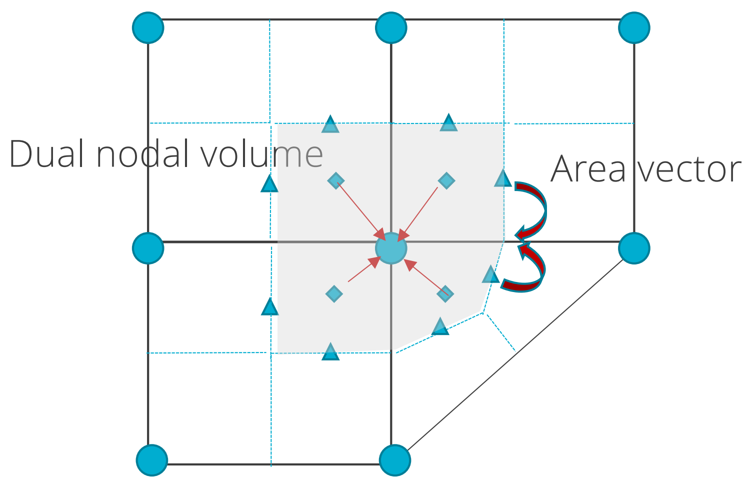
Cell-Centered (CC) and Edge-Based Vertex-Centered (EBVC): Each are generally a two-state scheme, Left (L) and Right(R)

- CC (A) and EBVC (B) – As shown below, each have a set of common features
  - Class Discussion: What are the common features?

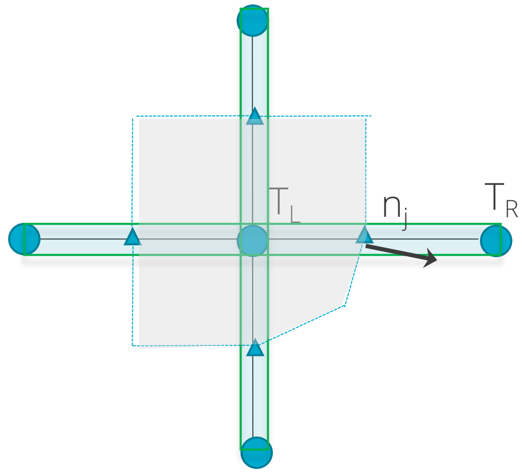


The Control Volume for EBVC is Defined by the ***Dual-Volume***

- All primitives are collocated at the vertices of the elements with equal-order interpolation
- A dual mesh is constructed to obtain flux and volume quadrature locations
- Classic two-state, “L” and “R” approach provides spatially second-order accuracy
- Iterate ***Nodes*** for volume-based contributions
- Iterate ***Edges*** for surface-based contributions



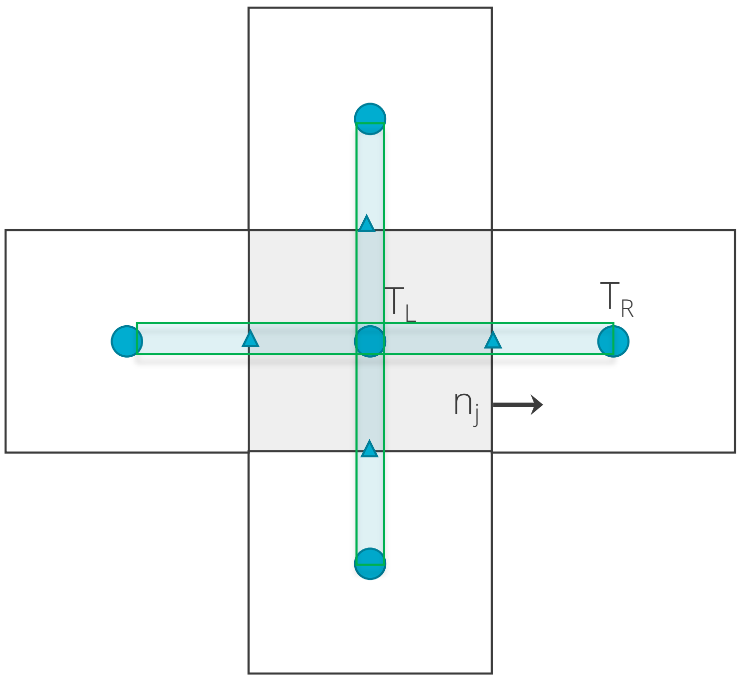
Dual-volume definition



Edge-based stencil

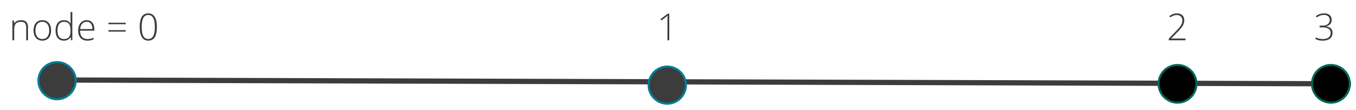
The Control Volume for CC is Defined by the ***Element Volume***

- All primitives are collocated at the cell-center of the element with equal-order interpolation
- Classic two-state, “L” and “R” approach provides spatially second-order accuracy
- Iterate ***Elements*** for volume-based contributions
- Iterate ***Virtual-Edges*** or ***Control Volume Faces*** for surface-based contributions



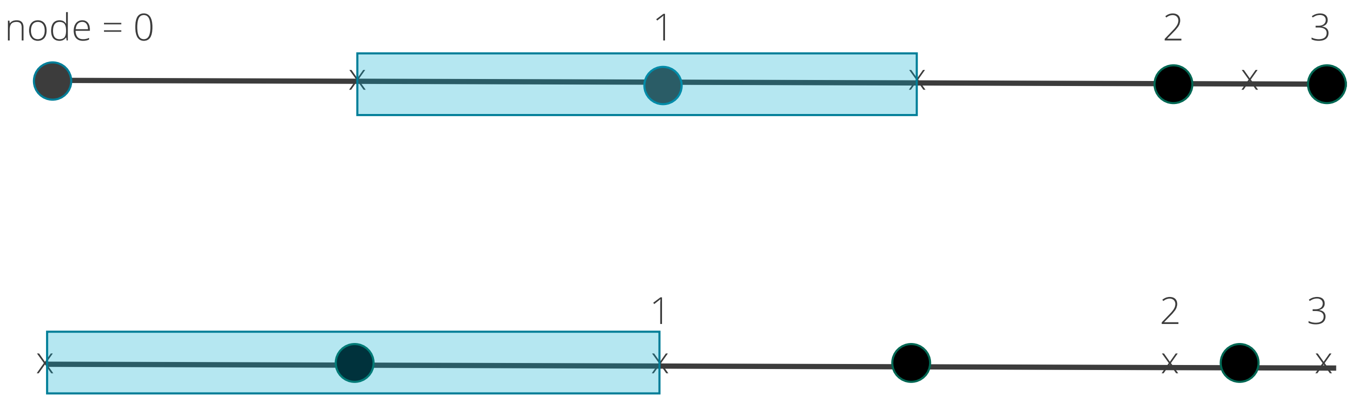
One-Dimensional Example of a Non-Uniform Mesh

- The nodes of the mesh need-not be equally spaced



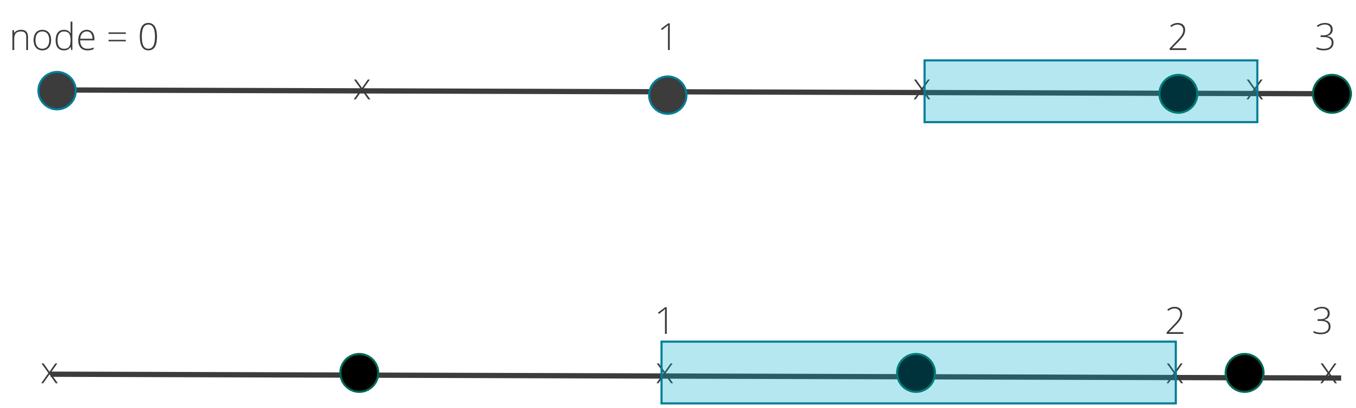
Control Volumes for CC and EBVC

- Note: Switched depiction of node (or vertex) to the cell-centered location



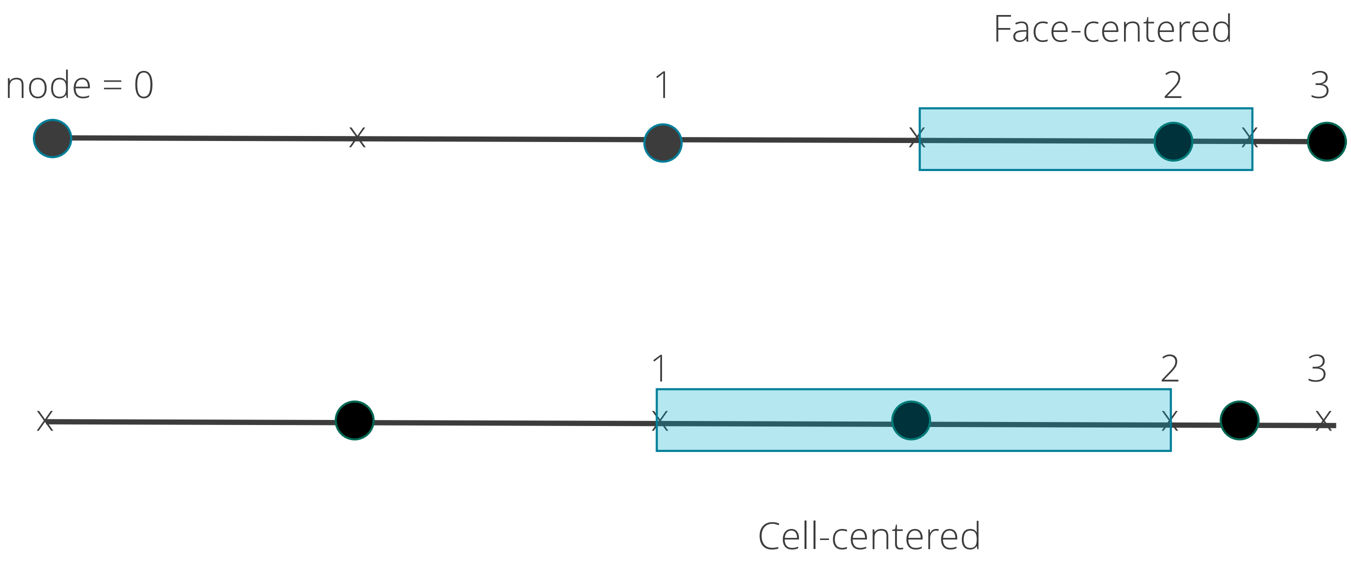
Control Volumes for CC and EBVC

- Note: Switched depiction of node (or vertex) to the cell-centered location



Control Volumes for CC and EBVC

- Face-centered vs Cell-Centered



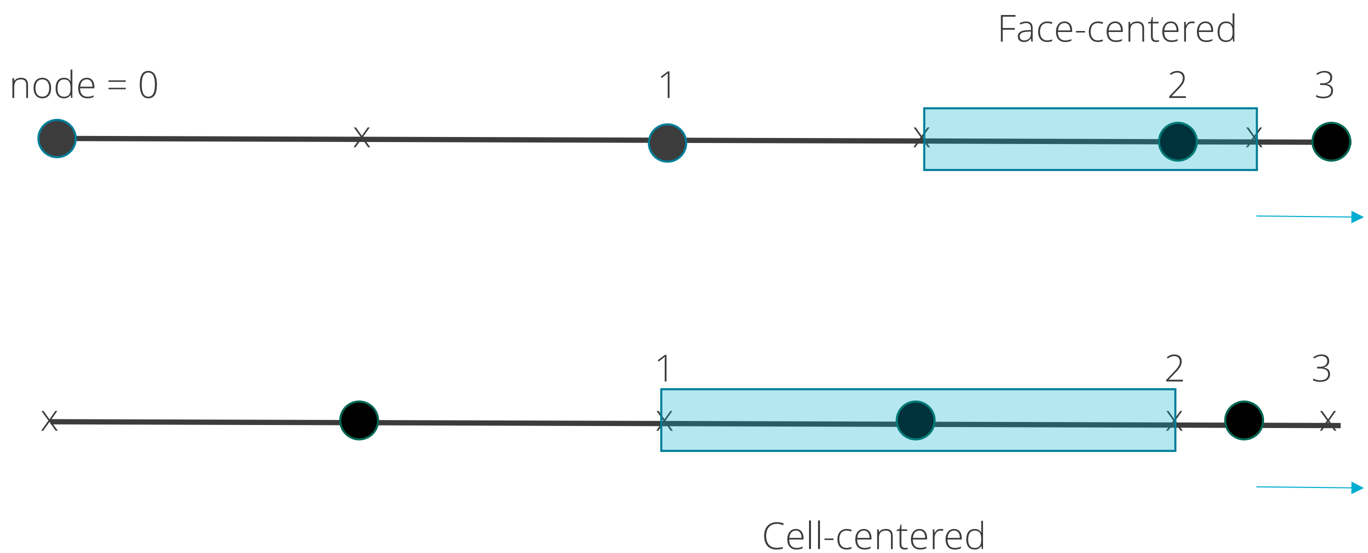
Basis Function Re-Visited

- Define an isoperimetric base element that varies from -1:+1, with integration point at the location ξ:

$$\phi_{ip} = \sum_n N_n^{ip} \phi_n$$

$$N_1 = \frac{(1-\xi)}{2}$$

$$N_2 = \frac{(1+\xi)}{2}$$



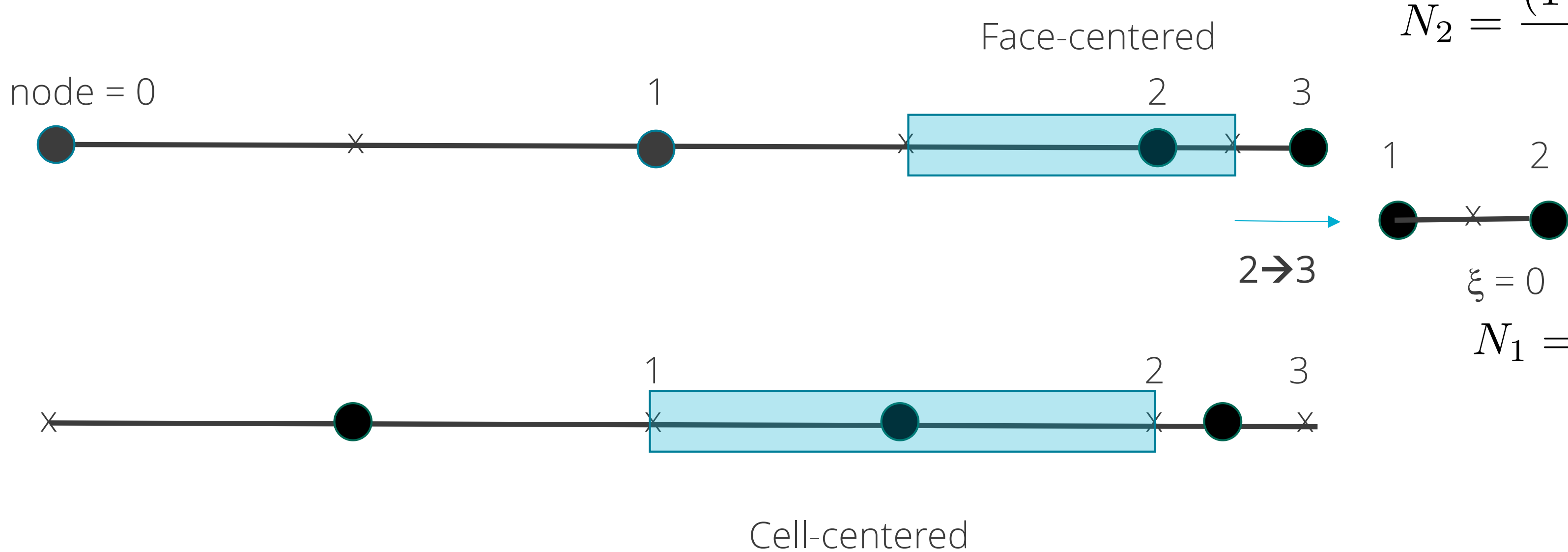
Basis Function Re-Visited

- Define an isoperimetric base element that varies from -1:+1, with integration point at the location ξ:

$$\phi_{ip} = \sum_n N_n^{ip} \phi_n$$

$$N_1 = \frac{(1-\xi)}{2}$$

$$N_2 = \frac{(1+\xi)}{2}$$



$$N_1 = N_2 = \frac{1}{2}$$

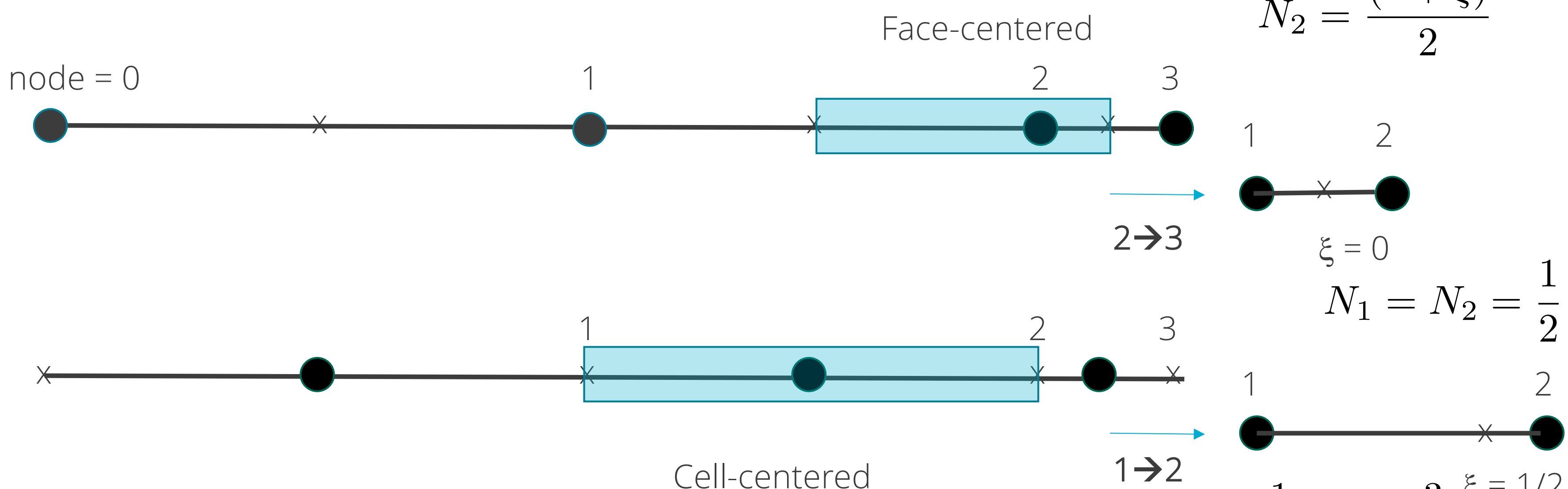
Basis Function Re-Visited

- Define an isoperimetric base element that varies from -1:+1, with integration point at the location ξ:

$$\phi_{ip} = \sum_n N_n^{ip} \phi_n$$

$$N_1 = \frac{(1-\xi)}{2}$$

$$N_2 = \frac{(1+\xi)}{2}$$



$$N_1 = N_2 = \frac{1}{2}$$

$$N_1 = \frac{1}{4}; N_2 = \frac{3}{4}$$

Other Examples of a Basis (This will become important for the element-based schemes)

Linear (Quad4)

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta)$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta)$$

$$N_3 = \frac{1}{4}(1+\xi)(1+\eta)$$

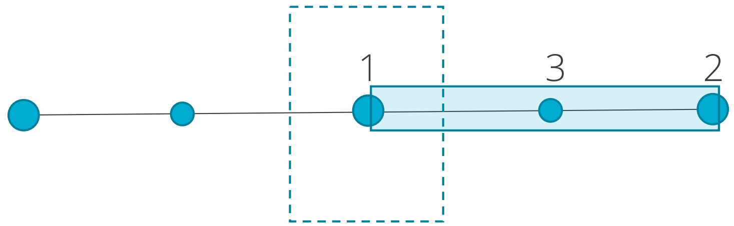
$$N_4 = \frac{1}{4}(1-\xi)(1+\eta)$$

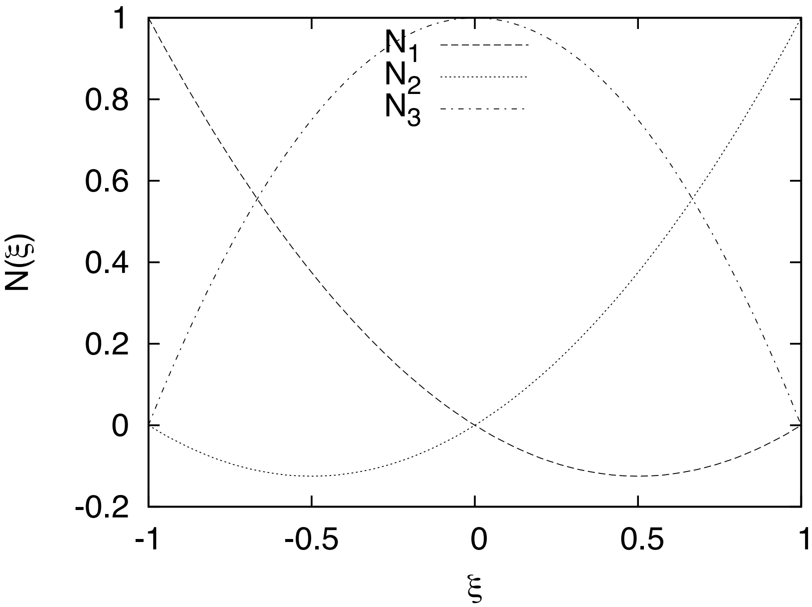
Quadratic (Bar3)

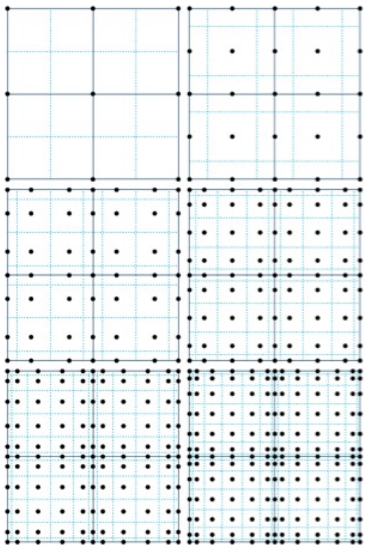
$$N_1 = \frac{-\xi(1-\xi)}{2}$$

$$N_2 = \frac{\xi(1+\xi)}{2}$$

$$N_3 = (1-\xi)(1+\xi)$$







Two-dimensional promotion

Simple One-D Diffusion Example

Let us introduce (briefly) the Method of Manufactured Solutions (MMS)

- Consider a simple diffusion equation:

$$\frac{d^2\phi}{dx^2} = 0$$

with a presumed, or manufactured solution: $\phi^{MMS}(x) = x^2$

$$\frac{d^2\phi^{MMS}}{dx^2} = S^{MMS} = 2$$

Gauss-Divergence

$$\int \frac{d^2\phi^h}{dx^2} dV = \int S^{MMS} dV \qquad \int \frac{d\phi^h}{dx} dS = \int S^{MMS} dV$$

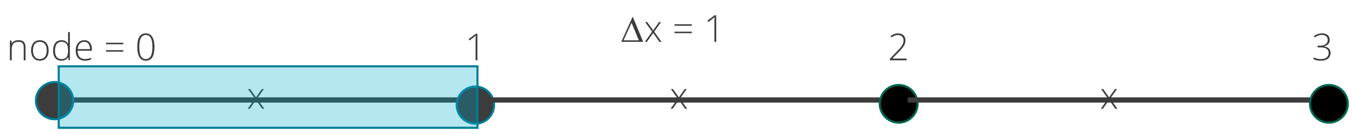
Discrete flux-based form at the sub-control volume surface (scs) and sub-control volume IP:

With error: $\epsilon = \phi^{MMS} - \phi^h$

$$\sum_{scsIP} \frac{d\phi^h}{dx} A_{scsIP} = \sum_{scvIP} S^{MMS} V_{scvIP}$$

Simple One-D MMS Example; Edge-based Assemble of Diffusion

We will use a linear basis and solve this system over a small patch of linear bar elements



Recall, simplified equation with RHS from source term already provided

$$\sum_{scsIP} \frac{d\phi^h}{dx}_{scsIP} = \sum_{scvIP} S^{MMS} \frac{\Delta x}{2}$$

Iterate elements with a simple left-hand side rule: += L and -=R

(note implicit conservation statement)

$$\frac{d\phi^h}{dx}_{scsIP} = \frac{\phi_1^h - \phi_0^h}{\Delta x} + C_0(\Delta x) + \ldots$$

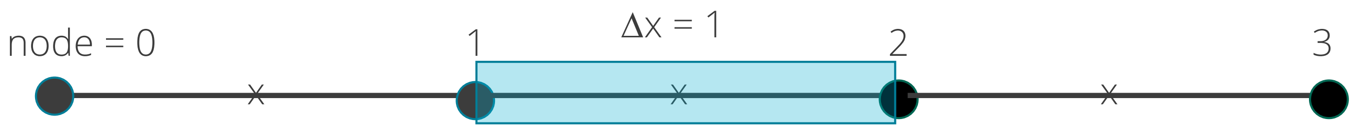
$$LHS_0 += \frac{\phi_1^h - \phi_0^h}{\Delta x}$$

$$LHS_1 -= \frac{\phi_1^h - \phi_0^h}{\Delta x}$$

Edge 1

Simple One-D MMS Example; Edge-based Assemble of Diffusion

We will use a linear basis and solve this system over a small patch of linear bar elements



Recall, simplified equation with RHS from source term already provided

$$\sum_{scsIP} \frac{d\phi^h}{dx}_{scsIP} = \sum_{scvIP} S^{MMS} \frac{\Delta x}{2}$$

Iterate elements with a simple left-hand side rule: += L and -=R

(note implicit conservation statement)

$$\frac{d\phi^h}{dx}_{scsIP} = \frac{\phi_2^h - \phi_1^h}{\Delta x} + C_0(\Delta x) + ...$$

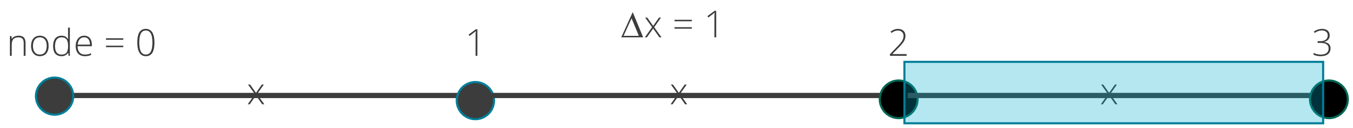
$$LHS_1 + = \frac{\phi_2^h - \phi_1^h}{\Delta x}$$

$$LHS_2 - = \frac{\phi_2^h - \phi_1^h}{\Delta x}$$

Edge 2

Simple One-D MMS Example; Edge-based Assemble of Diffusion

We will use a linear basis and solve this system over a small patch of linear bar elements



Recall, simplified equation with RHS from source term already provided

$$\sum_{scsIP} \frac{d\phi^h}{dx}_{scsIP} = \sum_{scvIP} S^{MMS} \frac{\Delta x}{2}$$

Iterate elements with a simple left-hand side rule: += L and -=R

(note implicit conservation statement)

$$\frac{d\phi^h}{dx}_{scsIP} = \frac{\phi_3^h - \phi_2^h}{\Delta x} + C_0(\Delta x) + \ldots$$

$$LHS_2 + = \frac{\phi_3^h - \phi_2^h}{\Delta x}$$

$$LHS_3 - = \frac{\phi_3^h - \phi_2^h}{\Delta x}$$

Edge 3

Simple One-D MMS Example; Collect all of the terms

$$LHS_0+ = \frac{\phi_1^h - \phi_0^h}{\Delta x}$$

$$LHS_1- = \frac{\phi_1^h - \phi_0^h}{\Delta x}$$

$$LHS_1+ = \frac{\phi_2^h - \phi_1^h}{\Delta x}$$

$$LHS_2- = \frac{\phi_2^h - \phi_1^h}{\Delta x}$$

$$LHS_2+ = \frac{\phi_3^h - \phi_2^h}{\Delta x}$$

$$LHS_3- = \frac{\phi_3^h - \phi_2^h}{\Delta x}$$

$$A = \frac{1}{\Delta x}\begin{bmatrix} -1 & +1 & 0 & 0 \\ +1 & -2 & +1 & 0 \\ 0 & +1 & -2 & 1 \\ 0 & 0 & +1 & -1 \end{bmatrix}$$

Fully assembled matrix

$$\frac{1}{\Delta x}\begin{bmatrix} -1 & +1 & 0 & 0 \\ +1 & -2 & +1 & 0 \\ 0 & +1 & -2 & 1 \\ 0 & 0 & +1 & -1 \end{bmatrix}\begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$

Fully assembled system

$$\frac{1}{\Delta x}\begin{bmatrix} +1 & 0 & 0 & 0 \\ +1 & -2 & +1 & 0 \\ 0 & +1 & -2 & 1 \\ 0 & 0 & 0 & +1 \end{bmatrix}\begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 9 \end{bmatrix}$$

Correct for BCs

$$\phi^h = \begin{bmatrix} 0 \\ 1 \\ 4 \\ 9 \end{bmatrix}$$

Solution is exact….. Linear basis exactly captures a quadratic solution

Model Passive Scalar Transport: Introducing the Test Function, w

- Time/Src/Advection/Diffusion equation for scalar ϕ

$$\frac{\partial \rho \phi}{\partial t} + \frac{\partial \rho u_j \phi}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\rho D \frac{\partial \phi}{\partial x_j}\right) = S^{\phi}$$

Under the assumption, $\rho D = \frac{\mu}{Sc}$ yields: $\frac{\partial \rho \phi}{\partial t} + \frac{\partial \rho u_j \phi}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\frac{\mu}{Sc} \frac{\partial \phi}{\partial x_j}\right) = S^{\phi}$

We can also integrate this PDE over the volume, while introducing a test function, w

$$\int w \left(\frac{\rho \phi}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_j \phi - \frac{\mu}{Sc}\frac{\partial \phi}{\partial x_j}\right) - S^{\phi}\right) dV = \int w R(\phi) dV$$

- The above is the so-called weak-form of the original (strong form) partial differential equation
- The elegant concept of a weighted residual statement is to seek a weak solution such that the residual is orthogonal to a selected space of weight functions

Model Passive Scalar Transport: Integration-by-Parts

- Recall, integration-by-parts, for example,

$$\int w \frac{\partial \rho u_j \phi}{\partial x_j} dV = - \int \rho u_j \phi \frac{\partial w}{\partial x_j} dV + \int w \rho u_j \phi n_j dS$$

- Note, that w must be differentiable at least once
- Under the special designation of w to be a piecewise constant, or Heaviside function, the gradient of this is a (negative) Diract-delta function

- A key property of this function is that the gradient of the piece-wise constant test function is a distribution of delta functions on the sub-control volume surface yielding,

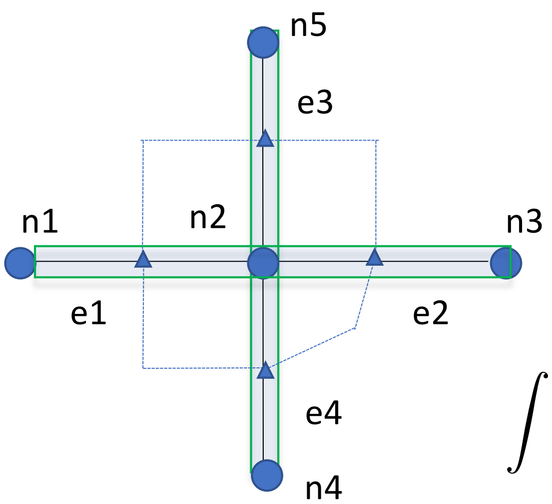
$$\int \left(\frac{\partial \rho \phi}{\partial t} - S^{\phi} \right) dV + \int \left(\rho u_j \phi - \frac{\mu}{Sc} \frac{\partial \phi}{\partial x_j} \right) n_j dS = 0$$

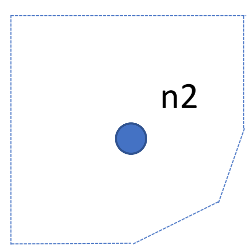
- Recall, we could have arrived at this form by simply usage of Gauss-Divergence, however, this formality allows us to view a finite volume scheme as a Petrov-Galerkin scheme, i.e., the underlying basis differs from the test function

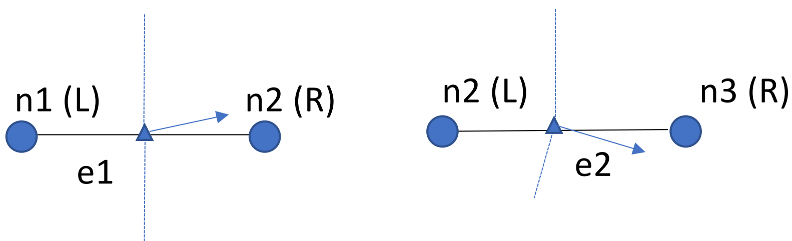
Deep Dive on EBVC: Node and Edge-loops

EBVC is a discretization scheme that:

- Uses a piecewise-constant test function
- Is a two-state flux approach, with a face-centered arrangement
- Iterates over locally-owned nodes for Time/Source/etc. (volumetric-based terms)
- Iterates over locally-owned edges for Advection/Diffusion terms (surface-based terms)







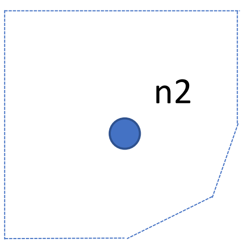
$$\int w \frac{\partial \rho u_j \phi}{\partial x_j} dV \approx \sum_{ip} (\rho u_j)_{ip} \phi_{ip} n j dS \approx \sum_{ip} \dot{m}_{ip} \phi_{ip}$$

$$\int w S^{\phi} dV \approx \sum_{nd} S^{\phi}_{nd} V_{nd}$$

EBVC: Implicit Time Discretization

- Let us define a general implicit time integrator that is A-stable
  - For y’ = k y ; y(0) = 1; y(t) = e^{kt} solution approaches zero as time increases for k < 0
- Backward Euler (two state) and is first-order accurate (A-stable)
- BDF2 (three state) is second-order accurate (A-stable)

This term is assembled over a nodal iteration and uses a single integration point (known as mass-lumping)



$$\int w \frac{\partial \rho \phi}{\partial t} dV \approx \sum_{nd} \frac{\left(\gamma_1 \rho_{nd}^{n+1} + \gamma_1 \rho_{nd}^{n} + \gamma_1 \rho_{nd}^{n-1}\right)}{\Delta t} V_{nd}$$

For uniform time stepping, the coefficients are give by, γ_1 = 3/2 γ_2 = -2 γ_3 = 1/2

EBVC: Implicit Time Discretization; Code

- https://github.com/NaluCFD/Nalu/blob/master/src/ScalarMassBDF2NodeSuppAlg.C

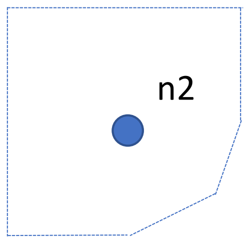
```
//--------------------------------------------------------------------------
//-------- node_execute ----------------------------------------------------
//--------------------------------------------------------------------------
void
ScalarMassBDF2NodeSuppAlg::node_execute(
  double *lhs,
  double *rhs,
  stk::mesh::Entity node)
{
  // deal with lumped mass matrix
  const double qNm1       = *stk::mesh::field_data(*scalarQNm1_, node);
  const double qN         = *stk::mesh::field_data(*scalarQN_, node);
  const double qNp1       = *stk::mesh::field_data(*scalarQNp1_, node);
  const double rhoNm1     = *stk::mesh::field_data(*densityNm1_, node);
  const double rhoN       = *stk::mesh::field_data(*densityN_, node);
  const double rhoNp1     = *stk::mesh::field_data(*densityNp1_, node);
  const double dualVolume = *stk::mesh::field_data(*dualNodalVolume_, node);
  const double lhsTime    = gamma1_*rhoNp1*dualVolume/dt_;
  rhs[0] -= (gamma1_*rhoNp1*qNp1 + gamma2_*qN*rhoN + gamma3_*qNm1*rhoNm1)*dualVolume/dt_;
  lhs[0] += lhsTime;
}
```

EBVC: Source Term Discretization

- Source terms for the edge-based scheme are also assembled over a nodal loop
- Note that in this scheme, we are using single point quadrature, i.e., the function is evaluated at a single point

The single point integration point leverages the fac that: p = 2N -1

- Where N is the number of integration points and p is the polynomial order
- For a linear basis, using one-point quadrature is design-order, or second-order in space



$$\int wS^{\phi}dV \approx \sum_{nd} S^{\phi}_{nd}V_{nd}$$

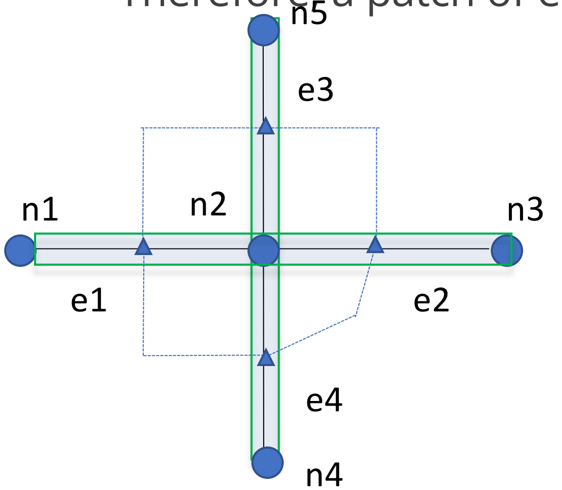
EBVC: Source Term Discretization; Code

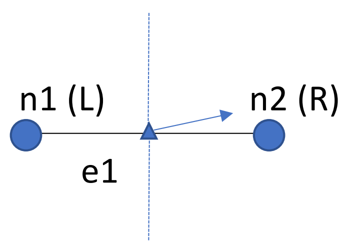
- https://github.com/NaluCFD/Nalu/blob/master/src/user_functions/VariableDensityMixFracSrcNodeSuppAlg.C

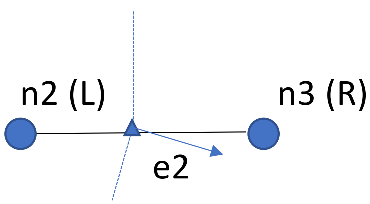
```
//--------------------------------------------------------------------------
//-------- node_execute ----------------------------------------------------
//--------------------------------------------------------------------------
void
VariableDensityMixFracSrcNodeSuppAlg::node_execute(
  double */*lhs*/,
  double *rhs,
  stk::mesh::Entity node)
{
  // deal with lumped mass matrix
  const double *coords = stk::mesh::field_data(*coordinates_, node);
  const double dualVolume = *stk::mesh::field_data(*dualNodalVolume_, node );
  const double x = coords[0];
  const double y = coords[1];
  const double z = coords[2];

  const double src = 0.10e1 * pow(znot_ * cos(amf_ * pi_ * x) * cos(amf_ * pi_ * y) * cos(amf_

  rhs[0] += src*dualVolume;
}
```

EBVC: Advection Term Discretization

- For advection, we have transformed the volume integral to a surface integration
- Therefore, a patch of edges are required for the full assembly at node 2







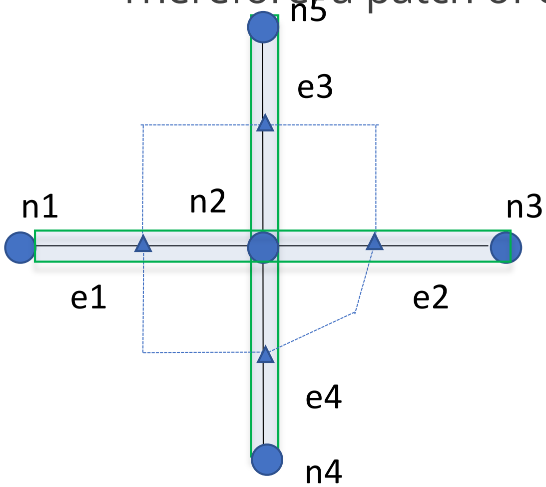
$$\int w \frac{\partial \rho u_j \phi}{\partial x_j} dV \approx \sum_{ip} (\rho u_j)_{ip} \phi_{ip} nj dS \approx \sum_{ip} \dot{m}_{ip} \phi_{ip}$$

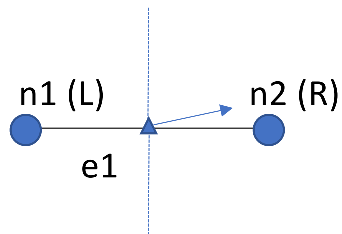
- Recall, that the mass flow rate at an integration point is prescribed
- Moreover, since the integration point is at the edge mid-point, $\phi_{ip} = (\phi^R + \phi^L)/2$

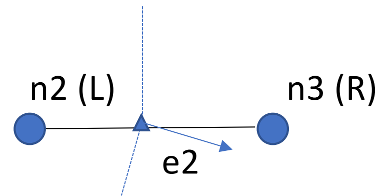
This is a *central*- or *Galerkin-based* advection operator; shown in conservative form, i.e, is in divergence form and has been integrated-by-parts

EBVC: Advection Term Discretization

- For advection, we have transformed the volume integral to a surface integration
- Therefore, a patch of edges are required for the full assembly at node 2







$$\int w \frac{\partial \rho u_j \phi}{\partial x_j} dV \approx \sum_{ip} (\rho u_j)_{ip} \phi_{ip} n j dS \approx \sum_{ip} \dot{m}_{ip} \phi_{ip}$$

- Recall, that the mass flow rate at an integration point is prescribed
- Moreover, since the integration point is at the edge mid-point, $\phi_{ip} = (\phi^R + \phi^L)/2$

For kinetic energy conservation, a CC scheme also uses ½ weights!

This is a *central-* or *Galerkin-based* advection operator; shown in conservative form, i.e, is in divergence form and has been integrated-by-parts

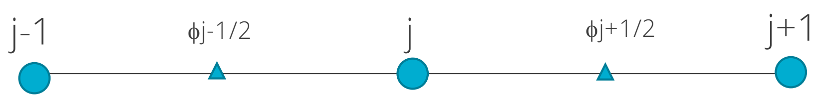
Mahesh et al., JCP (2004)

EBVC: Advection Term Discretization; Code

- https://github.com/NaluCFD/Nalu/blob/master/src/AssembleScalarEdgeSolverAlgorithm.C

Upwind Advection Operators

- Recall from our finite difference class that in absence of a stabilizing diffusion term, or when the velocity is very high, i.e., the Peclet number > 2, non-physical oscillations can be noted
- An upwind approach was devised to mitigate this at the expense of numerical accuracy
- For a finite volume approach, upwinding is very easy to construct



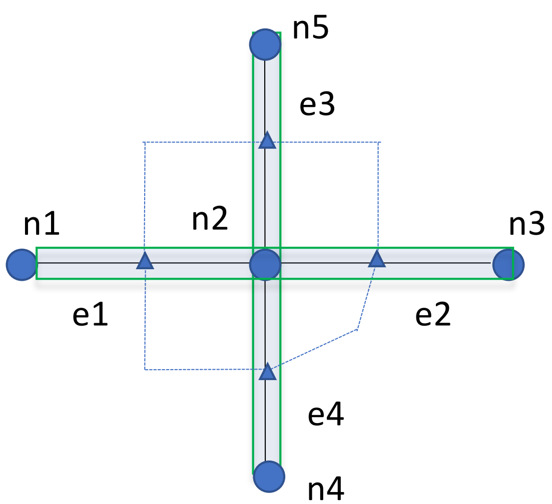
$$\phi_{j+1/2} = \begin{cases} \phi_j, & \text{if } u_l n_l > 0. \\ \phi_{j+1}, & \text{otherwise.} \end{cases}$$

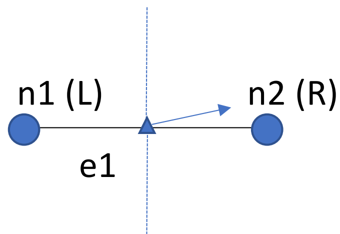
"as the wind blows"

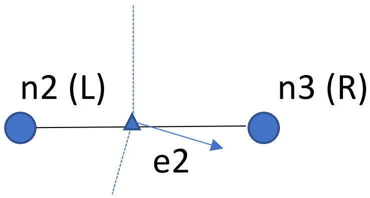
- However, upwind operators, even if higher-order, are numerically dissipative due to loss of symmetry of the operator
- Higher-order upwind (second-order in space) can be obtained on an unstructured mesh by use of projected nodal gradients and extrapolation (will cover this in the advection operator section)

EBVC: Diffusion

- For diffusion, we have transformed the volume integral to a surface integration
- Therefore, a patch of edges are required for the full assembly at node 2





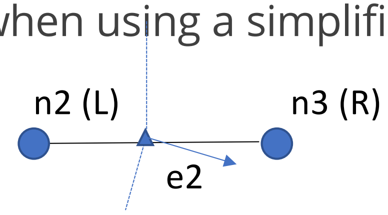


$$\int w \frac{\partial q_j}{\partial x_j} dV \approx -\sum_{ip} \left(\frac{\mu}{Sc}\right)_{ip} A_j^{ip}$$

EBVC: Diffusion, Non-Orthogonality

- The edge-based diffusion operator is a bit more complex in that we are computing a gradient at the integration point while using two nodes, the left and right
- In many practical meshes, the distance vector between the edges (dx_j) and the edge area vector will not be perfectly orthogonal
- This non-orthogonality causes inaccuracy when using a simplified approach

$$\frac{\partial \phi}{\partial x_j} = \frac{(\phi_R - \phi_L)}{|\Delta x_l|} n_j$$



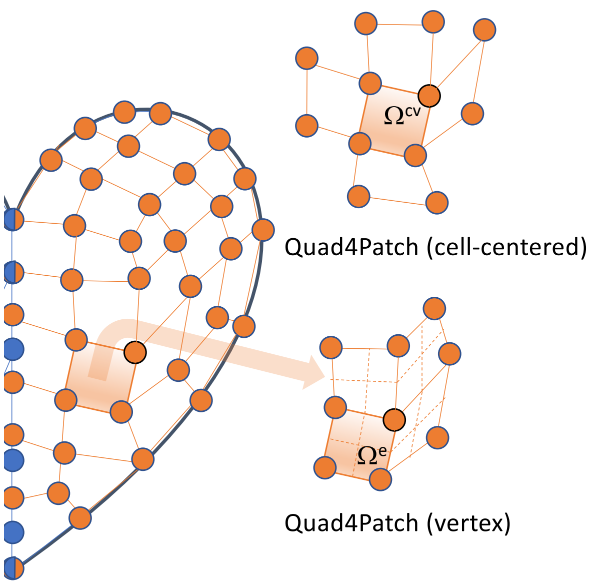
- The non-orthogonality approach of Jasek (Ph.D. Imperial College) is a standard cell-centered approach that has been adopted for edge-based schemes
- CC and EBVC are each two-state schemes and share non-orthogonality issues

$$\left.\frac{\partial \phi}{\partial x_j}\right|_{ip} = \overline{G_j \phi} + \left[(\phi_R - \phi_L) - \overline{G_l \phi} \Delta x_l \right] \frac{A_j}{A_k \Delta x_k}$$

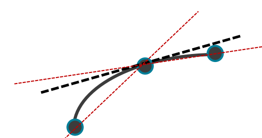
Where $G_j\phi$ is a "projected nodal gradient"

(Projected) Nodal Gradient Operator

- Recall, that the edge-based diffusion operator, and as we will see in future lectures for some choices for the advection operator, a nodal gradient is required



- First, the gradient of a function (other than linear) is discontinuous, i.e., the value at a shared element face depends on which element is used to compute the gradient



- Therefore, we can view the nodal gradient a continuous at the nodes and discontinuous within the elements

Let's minimize this difference: $\frac{1}{2}\left(\frac{\partial \phi}{\partial x_j} - G_j \phi\right)^2$

by solving:

$$\int w G_j \phi dV = \int w \frac{\partial \phi}{\partial x_j} dV \longrightarrow G_j \phi = \frac{\sum_{ip} \phi_{ip} n_j dS}{V}$$

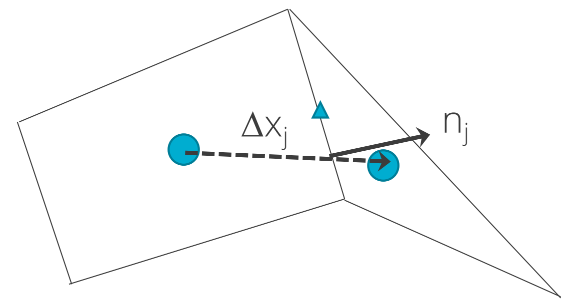
EBVC: Diffusion, Non-Orthogonality; Code

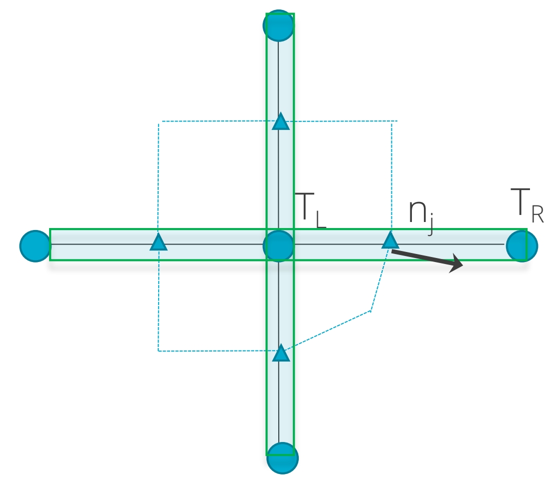
- https://github.com/NaluCFD/Nalu/blob/master/src/AssembleScalarEdgeDiffSolverAlgorithm.C

$$
\begin{aligned}
-\int w \frac{\mu}{Sc} \frac{\partial \phi}{\partial x_j} dV \approx \quad & -\frac{\mu}{Sc}\Big|_{ip} \Bigg[\left(\overline{G_x \phi} A_x + \overline{G_y \phi} A_y\right) + (\phi_R - \phi_L) \frac{A_x A_x + A_y A_y}{A_x \Delta x_x + A_y \Delta x_y} \\
& - \left(\overline{G_x \phi} dx + \overline{G_y \phi} dy\right) \frac{A_x A_x + A_y A_y}{A_x dx + A_y dy} \Bigg]
\end{aligned}
$$

Typical Difficulties for a Two-State Scheme

- With two points, only a linear basis can be used, therefore, unstructured CC and EBVC are limited to second-order spatial accuracy
- Non-orthogonality, or the mis-alignment of the distance vector between the two ”L” and “R” states and the surface normal, is present
- Integration point
- Both edge- and cell centered-based schemes show degraded accuracy on typical production meshes
- Several non-orthogonality approaches are available, for the best source, see Jasak
  - Jasak, “Error analysis and error estimation for the finite volume method with applications to fluid flow”, Imperial College Dissertation, 1996





For Instance, Verification of The Diffusion Operator